DOI：10.3901/JME.2019.16.220

基于块结构性质的花粉算法求解可重入作业车间调度问题*

孙在省 钱 斌 胡 蓉 张梓琪 张长胜
(昆明理工大学信息工程与自动化学院 昆明 650500)

摘要：针对可重入作业车间调度问题(Reentrant job shop scheduling problem, RJSSP)，提出一种基于块结构性质的花粉算法(Flower pollination algorithm based on block structure properties, FPA_BSP)，用于最小化总加权延误时间(Total weighted tardiness, TWT)。首先，建立 RJSSP 基于析取图的数学模型，并证明在确定析取弧方向后，该模型的对偶模型为最大费用流问题模型。其次，设计扩展 RSOV (Reentrant-smallest-order-value, RSOV)编码规则，将花粉算法的实数矢量个体转变为排列矢量，使其可对问题解空间进行全局搜索，以发现存在优质解的区域。然后，定义 8 种邻域结构，并基于最大费用流问题特性分析块结构内部性质，得到前 4 种邻域结构能改进 TWT 的判定条件，可用于避免对无效区域的搜索，进而提出融合多种邻域的高效局部搜索，对全局搜索发现的优质解区域进行细致搜索。试验和算法比较验证 FPA_BSP 的有效性。提出 RJSSP 的块结构性质，并将其与花粉算法结合得到求解 RJSSP 的有效算法 FPA_BSP，首次将花粉算法用于求解车间调度问题。

关键词：可重入作业车间调度问题；总加权延误时间；花粉算法；块结构性质
中图分类号：TG156

Flower Pollination Algorithm Based on Block Structure Properties for Reentrant Job Shop Scheduling Problem

SUN Zaixing QIAN Bin HU Rong ZHANG Ziqi ZHANG Changsheng
(College of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500)

Abstract：Flower pollination algorithm based on block structure properties (FPA_BSP) is presented to minimize the total weighted tardiness (TWT) for reentrant job shop scheduling problem (RJSSP). Firstly, the mathematical model of RJSSP based on the disjunctive graph is established, and then its duality model is proved to be the model of maximum cost flow problem after it being determined the direction of the disjunctive arcs. Secondly, the extended reentrant-smallest-order-value (RSOV) encoding rule is designed to transform FPA's individuals from real vectors to job permutations so that flower pollination algorithm (FPA) can be used to perform global search for finding high-quality solutions or regions in the solution space. Thirdly, eight kinds of neighborhood structures are defined, and the inner properties of block structures are analyzed based upon the properties of the maximum cost flow problem. The determination conditions of the four former neighborhood structures for improving TWT are obtained by the block structures' analyses, which can be used to avoid search in the invalid regions. Then, a high-efficient local search integrating multiple neighborhoods with the obtained determination conditions is proposed to execute a thorough search from the promising regions found by the global search. Computational experiments and comparisons manifest the effectiveness of the presented FPA_BSP. The properties of RJSSP's block structures and incorporates them with FPA are proposed to obtain an effective FPA_BSP for solving RJSSP. It is also the first time that FPA is used to address scheduling problem.

Key words：reentrant job shop scheduling problem；total weighted tardiness；flower pollination algorithm；block structure properties

* 国家自然科学基金(51665025，60904081)和云南省自然科学基金(2015FB136)资助项目。20181215 收到初稿；20190502 收到修改稿

0　前言

车间调度是机械加工车间进行生产加工的重要制造和决策问题。作业车间调度问题(Job shop scheduling problem, JSSP)是最典型最困难的组合优化问题之一[1]，GAREY 已证当机器数超过两台时 JSSP 是一个 NP-complete 问题[2]，它的算法研究一直是学术界和工程界共同关注的重要课题。JSSP 局限于零件的加工或制造只能在一台机器上加工一次，而在半导体制造、计算机网络等行业的生产系统中，部分零件要求被同一台机器加工多次，故可重入作业车间调度问题(Reentrant job shop scheduling problem, RJSSP)的研究更符合实际的生产环境，具有重要的工程价值和现实意义。近年来，在面向现代生产的供应链系统中及时交货是非常重要的，避免延误的目标越来越受到重视。因此，本文针对以最小化总加权延误时间(Total weighted tardiness, TWT)这一正规性指标[3]为优化目标的 RJSSP，进行有效求解算法研究。由于 JSSP 具有 NP-complete 属性，而 JSSP 可归约(Reduce to)于 RJSSP，故可得 RJSSP 也属于 NP-complete 问题。因此，对最小化 TWT 的 RJSSP 的求解算法研究具有较高实际和理论价值，可为相关生产企业提供切实的指导和帮助。

就 JSSP 而言，搜索邻域结构对其求解算法的性能影响至关重要，同时也有一定的通用性，故研究意义重大。已有学者对以最小化最大完工时间为目标的 JSSP 的邻域结构进行了研究。1988 年，VAN 等[4]在 BALAS[5]的基础上提出的邻域结构 N1，为目前许多有效搜索策略奠定了基础，但这种邻域范围相当大且包括大量的无效移动。1988 年，MATSUO 等[6]提出邻域结构 N2，并证明交换块内的相邻工序不可能缩短最大完工时间。与其他有效算法的仿真比较验证了基于 N2 的算法的优越性。1988 年，GRABOWSKI 等[7-8]扩展了其早期为一台机器定义的邻域结构，首次引入了块结构的概念。1993 年，DELL'AMICO 等[9]提出两种类型的邻域结构 N4，其基于 N4 邻域结构的禁忌搜索算法在一段时间内曾是求解 JSSP 最强方法。1996 年，NOWICKI 等[10]提出邻域结构 N5，N5 由于过多的限制了无效移动，在某种程度上降低了搜索效率。1998 年，BALAS 等[11]提出邻域结构 N6，通过仿真比较验证了所设计的基于 N6 的算法性能最优。2007 年，ZHANG 等[12]提出邻域结构 N7，通过仿真试验证明所设计基于 N7 的禁忌搜索算法优于其他几种算法。近年来，以最小化 TWT 为目标的 JSSP 的邻域结构也得到一定研究。2011 年，ZHANG 等[13]在求解以最小化 TWT 为目标的 JSSP 提出的邻域结构中，利用该问题的对偶模型的性质，证明了所提邻域结构在满足某些约束判定条件时不能改进目标值，有效增强了算法的局部搜索效率。2016 年，KUHPFAHL 等[14]提出的六种新的邻域结构，仿真试验比较了不同结构及其组合的性能和搜索能力。2016 年，赵诗奎[15]在求解以最小化最大完工时间为目标的 JSSP 中，通过对机器空闲时间的直接和间接利用，设计新的邻域结构，提出一种混合算法，结合基准算例测试分析，验证了所提算法具有良好的求解性能。通过采用合适的邻域结构，上述文献中的算法均取得了良好性能。然而，目前尚无对 RJSSP 这一重要问题的有效邻域结构的相关研究。

对于 RJSSP，2009 年，TOPALOGLU 等[16]提出转移瓶颈算法，用于求解以最小化最大完工时间为目标的 RJSSP，通过与其他算法的仿真比较，表明所提算法是非常有效的。2013 年，QIAN 等[17]提出混合差分进化算法，用于求解以机器总空闲时间和最大延误时间为评价指标的多目标 RJSSP，仿真试验和对比验证了所提算法的有效性。2014 年，CHEN 等[18]基于贝叶斯统计推断的分布估计算法用于求解以最小化最大完工时间为目标的带序相关设置时间的 RJSSP，利用概率模型产生新种群并指导算法的搜索方向，仿真试验和比较表明了该算法的有效性和鲁棒性。由文献调研可知，RJSSP 的求解算法已开始得到初步研究，但对以最小化 TWT 为目标的 RJSSP，尚未有人进行有效求解算法研究。

受自然界中开花植物花朵授粉过程的启发，英国剑桥大学学者 YANG[19]在 2012 年根据开花植物的授粉过程的特征、花的稳定性和传粉者的行为理想化等特点首次提出的一种新型元启发式群智能优化算法——花粉算法(Flower pollination algorithm, FPA)，在处理连续优化问题时得到很好的效果。该算法融合了布谷鸟算法和蝙蝠算法可进行全局分散搜索的优点，是一种非常有应用前景的算法，目前尚无其用于求解车间调度问题的文献报道，故进行相关研究非常重要。

本文提出一种基于块结构性质的花粉算法(Flower pollination algorithm based on block structure properties, FPA_BSP)，用于求解以最小化 TWT 为目标的 RJSSP(简称 TWT-RJSSP)。首先，在 JSSP 的析取图模型上加入自定义的重入弧，建立 RJSSP 基于析取图的数学规划模型，并证明在进一步确定析取弧方向后，该模型的对偶模型即为最大费用流问

题模型。其次，设计扩展的 RSOV (Reentrant-smallest-order-value, RSOV)编码规则，将 FPA 用于在问题解空间执行全局搜索，以发现存在优质解的区域。然后，给出 8 种邻域结构，并基于最大费用流问题特性分析问题关键路径上的块结构内部性质，得到前 4 种邻域结构能否改进优化目标的判定条件, 可用于排除对解空间中部分无效区域的搜索，同时采用后 4 种邻域结构适当扩展搜索区域，进而提出融合多种邻域的高效局部搜索，对全局搜索发现的优质解区域进行细致搜索。仿真试验和算法比较表明 FPA_BSP 可有效求解 TWT-RJSSP。

1 RJSSP 的问题描述

RJSSP 的描述如下：考虑 n 个工件 $J=\{1,2,\cdots,n\}$，在 m 台机器 $M=\{1,2,\cdots,m\}$ 上加工，且每个工件在每台机器上可重复加工 r 次的过程。根据工件加工工艺要求，已知每个工件延误权重 ω_j、交货期 μ_j、各工序的加工时间及各工件在各机器上的加工次序约束，每个工件按其约束在不同的时间可重复访问同一台机器。同时假设：工序的加工过程不允许被中断；同一时刻一个工件至多在一台机器上加工，同一时刻一台机器至多加工一道工序；每道工序必须在前一道工序加工完毕之后方可开始加工；机器之间的缓冲区容量无限大，忽略不同机器间的工件运输时间和在同一台机器上不同操作间的设置时间。本文的 RJSSP 就是如何安排每台设备上工序的加工顺序，使得 TWT 最小。

1.1 RJSSP 的析取图

析取图由 ROY 和 SUSSMAN 于 1964 年提出，描述调度问题中工艺顺序约束和机器唯一性约束。RJSSP 可以用析取图 $G=(N,A,E,Re)$ 来表示。其中，$O=\{1,2,\cdots,TO\}$ 为 RJSSP 的所有工序的集合；N 为工序节点的集合，$N=O\cup\{0\}=\{0,1,\cdots,TO\}$，其中 0 为虚拟节点；$F(O)$ 为每个工件的第一个工序的集合；$U(O)$ 为每个工件的最后一个工序的集合；$R(O)$ 为重入工序的集合；$TO=n\times m\times r$ 为总的工序数，虚拟工序 0 的开始加工时间和加工时间设定为零；A 为合取弧集合，合取弧描述同一个工件工艺路线的顺序约束，为单向；U 为尾弧，描述每个工件最后一个工序指向虚拟节点 0，为单向；E 为析取弧集合，描述在同一台机器上加工工序的前后关系，在调度之前析取弧取向未定，为双向；Re 是针对可重入特性首次提出的重入弧集合，描述同一个工件在同一台机器上的加工约束，为单向。析取弧取向确定的过程，即为调度问题的求解过程。

图 1 为一个 $n\times m\times r=2\times2\times2$ 的 RJSSP 中一台机器上的析取图(图 1 中，①、②、③、④为工件 1 的工序；⑤、⑥、⑦、⑧为工件 2 的工序)。表 1 为该实例的具体数据。

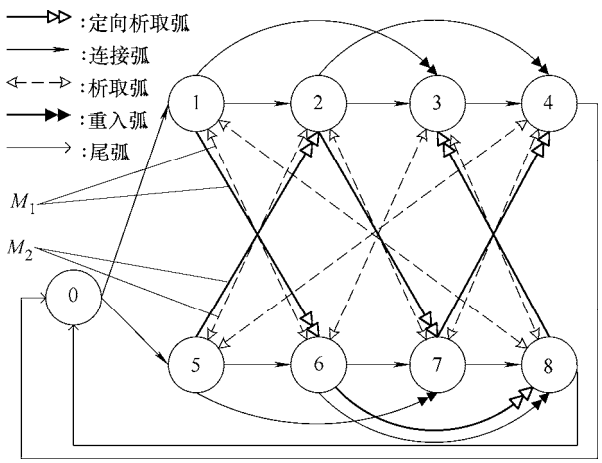

图 1 $2\times2\times2$ 的 RJSSP 的析取图

表 1 $2\times2\times2$ 的 RJSSP(“ — ”表示不在相应的机器上加工)

工件	工序号	加工机器及时间		延误权重	交货期
		M_1	M_2		
J_1	1	5	—	5	152
	2	—	15		
	3	50	—		
	4	—	56		
J_2	5	—	42	9	213
	6	75	—		
	7	—	34		
	8	25	—		

由图 1 和表 1 可知, 工件1有四个工序$\{1,2,3,4\}$，工件 2 有四个工序$\{5,6,7,8\}$，每个工件必须按工序号的顺序来加工，其中 $O=\{1,2,3,\cdots,7,8\}$，$N=O\cup\{0\}$，$F(O)=\{1,5\}$，$U(O)=\{4,8\}$，$R(O)=\{3,4,7,8\}$，$A=\begin{Bmatrix}(0,1),(1,2),(2,3),(3,4)\\(0,5),(5,6),(6,7),(7,8)\end{Bmatrix}$，$U=\begin{Bmatrix}(4,0)\\(8,0)\end{Bmatrix}$，$E=\begin{Bmatrix}\langle1,6\rangle,\langle1,8\rangle,\langle3,6\rangle,\langle3,8\rangle\\\langle2,5\rangle,\langle2,7\rangle,\langle4,5\rangle,\langle4,7\rangle\end{Bmatrix}$，$Re=\begin{Bmatrix}(1,3),(2,4)\\(5,7),(6,8)\end{Bmatrix}$。合取弧 A 与重入弧 Re 里面的元素 (i,j) 有加工约束，必须先加工 i 后加工 j，而析取弧 E 中的元素 $\langle i,j\rangle$ 无加工约束限制。当机器 M_1 上的析取弧方向确定, 可得机器 M_1 上的定向析取弧集合$\{(1,6),(6,8),(8,3)\}$，如图 1 中单向实线双空心箭头, 同时也就确定了机器 M_1 上的操作加工顺序为 $[1,6,8,3]$。机器 M_2 上操作加工顺序的确定和机器 M_1 上的类似。

1.2　RJSSP 的析取规划模型

令 s_i 为工序 i 的开始加工时间，其中 $s_0=0$；$\boldsymbol{S}$ 为所有工序的开始加工时间组成的矢量，即决策变量；p_i 为工序 i 的加工时间，其中 $p_0=0$；μ_j 为工件 j 的最后一个工序；(i_1,i_2) 为有加工约束的弧，即先加工 i_1，后加工 i_2；$\langle i_1,i_2\rangle$ 为方向未定的析取弧，“$\vee$”为逻辑或，描述析取弧中工序的关系；c_i、d_i 和 ω_i 分别为工序 i 所属工件的完工时间、交货期和延误权重；T_{μ_j} 为工件 j 的延误时间，$T_j=\max\{0,c_j-d_j\}$。

RJSSP 的析取规划模型描述如下

$$\min \quad TWT(\boldsymbol{S})=\sum_{\mu_j\in U(O)}\omega_{\mu_j}T_{\mu_j} \tag{1}$$

$$\text{s.t.}\quad s_{i_1}+p_{i_1}\le s_{i_2} \qquad \forall(i_1,i_2)\in A \tag{2}$$

$$\left(s_{i_1}+p_{i_1}\le s_{i_2}\right)\vee\left(s_{i_2}+p_{i_2}\le s_{i_1}\right)\ \forall\langle i_1,i_2\rangle\in E \tag{3}$$

$$s_{i_1}+p_{i_1}\le s_{i_2} \qquad \forall(i_1,i_2)\in Re \tag{4}$$

$$T_{\mu_j}\ge s_{\mu_j}+p_{\mu_j}-d_{\mu_j} \qquad \forall\mu_j\in U(O) \tag{5}$$

$$T_{\mu_j}\ge 0 \qquad \forall\mu_j\in U(O) \tag{6}$$

上述模型中，由 $s_0=p_0=0$ 和式(2)可得 $s_i\ge 0\ \ (\forall i\in O)$，故不需单独给出此约束。

1.3　确定析取弧方向后的线性规划模型

根据析取图的定义，求解 RSSP 等价于确定所有析取弧的方向，即在式(3)中 $\forall\langle i_1,i_2\rangle\in E$ 的两个条件只有一个符合要求。RSSP 的可行解在析取图中一定不包含有向环。当 1.2 节模型所有析取弧的方向确定之后，由于每个工件的完工时间不一定在交货期内，可能存在工件延误问题，故仍然需进一步优化求解。令 σ 为所有析取弧确定方向后得到的定向析取弧集合。这时问题的优化模型可以用线性规划模型来表示，具体如下

$$\min \quad TWT_\sigma(S)=\sum_{\mu_j\in U(O)}\omega_{\mu_j}T_{\mu_j} \tag{7}$$

$$\text{s.t.}\quad s_{i_2}-s_{i_1}\ge p_{i_1} \qquad \forall(i_1,i_2)\in A \tag{8}$$

$$s_{i_2}-s_{i_1}\ge p_{i_1} \qquad \forall(i_1,i_2)\in \sigma \tag{9}$$

$$s_{i_2}-s_{i_1}\ge p_{i_1} \qquad \forall(i_1,i_2)\in Re \tag{10}$$

$$T_{\mu_j}-s_{\mu_j}\ge p_{\mu_j}-d_{\mu_j} \qquad \forall\mu_j\in U(O) \tag{11}$$

$$T_{\mu_j}\ge 0 \qquad \forall\mu_j\in U(O) \tag{12}$$

本节模型与第 1.2 节模型的不同之处在于式(3)与式(9)，式(9)为确定析取弧方向后的约束。式(9)的确定使得第 1.2 节的析取规划模型转变为本节的线性规划模型。此外，本节模型中 $\forall(i_1,i_2)\in A\cup\sigma\cup Re$ 必须满足 $s_{i_2}-p_{i_1}\ge s_{i_1}$，此结构对于考虑其对偶问题十分重要[20]。

1.4　上一节模型的对偶模型

本节证明第 1.3 节线性规划问题模型的对偶模型是最大费用流问题模型。

最大费用流问题模型的定义如下

$$\max \quad TC_\sigma(F)=\sum_{(i_1,i_2)\in A\cup\sigma\cup Re}p_{i_1}F_{i_1,i_2}+\sum_{\mu_j\in U(O)}\left(p_{\mu_j}-d_{\mu_j}\right)F_{\mu_j,0} \tag{13}$$

$$\text{s.t.}\quad F_{\mu_j,0}\le\omega_{\mu_j} \qquad \forall\mu_j\in U(O) \tag{14}$$

$$\sum_{\xi:(\xi,i)\in A\cup\sigma\cup Re}F_{\xi,i}-\sum_{\xi:(i,\xi)\in A\cup\sigma\cup Re\cup U}F_{i,\xi}=0\ \forall i\in O \tag{15}$$

$$F_{i_1,i_2}\ge 0 \qquad \forall(i_1,i_2)\in A\cup\sigma\cup Re\cup U \tag{16}$$

在上述模型中，$F_{i_1,i_2}\ (i_1,i_2\in N)$ 为弧 (i_1,i_2) 上的流量；式(14)和式(15)为析取图 $G=(N,A,E,Re)$ 中的流量平衡约束[16]。在 $A\cup\sigma\cup Re$ 中的每一个弧有单位成本 $c_{i_1,i_2}=p_{i_1}$ 和无限容量，在 U 中的每一个弧有单位成本 $c_{\mu_j}=p_{\mu_j}-d_{\mu_j}$ 和有限容量 ω_{μ_j} (即式(14))。而式(15)要求流量平衡，即在 O 中的任意节点，总输入的流量应等于总输出的流量。

定理 1：第 1.3 节线性规划问题模型的对偶模型是最大费用流问题模型。

证明：将第 1.3 节线性规划问题模型写成如下的矩阵标准形式

$$\begin{aligned}&\min \quad \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}\\&\text{s.t.}\quad \boldsymbol{A}\boldsymbol{x}\ge\boldsymbol{b}\\&\qquad \boldsymbol{x}_1\ge\boldsymbol{0}\\&\qquad \boldsymbol{x}_2:\text{无限制}\end{aligned} \tag{17}$$

式中，$\boldsymbol{x}=(\boldsymbol{x}_1,\boldsymbol{x}_2)^{\mathrm{T}}$ 为所有决策变量的列矢量。令 $\boldsymbol{x}_1=(T_{\mu_1},T_{\mu_2},\cdots,T_{\mu_n})^{\mathrm{T}}$，$\boldsymbol{x}_2=(s_1,s_2,\cdots,s_{TO})^{\mathrm{T}}$，$\boldsymbol{c}^{\mathrm{T}}=(\boldsymbol{c}_1^{\mathrm{T}},\ \boldsymbol{c}_2^{\mathrm{T}})$，$\boldsymbol{c}_2^{\mathrm{T}}=\boldsymbol{0}_{1\times TO}$，$\boldsymbol{c}_1^{\mathrm{T}}=(\omega_{\mu_1},\omega_{\mu_2},\cdots,\omega_{\mu_n})$。显然，$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}=\boldsymbol{c}_1^{\mathrm{T}}\boldsymbol{x}_1+\boldsymbol{c}_2^{\mathrm{T}}\boldsymbol{x}_2$。

由析取图的性质可知，$|A|=n+n(mr-1)=mnr$ 为合取弧的个数；$|\sigma|=m(nr-1)$ 为定向析取弧的个数；$|Re|=mn(r-1)$ 为重入弧的个数；令 $\mu=|A\cup\sigma\cup Re|=|A|+|\sigma|+|Re|=3mnr-mn-m$。注意到 1.3 节模型的约束可以分为(8,9,10)和(11)两种形式，因此可将矩阵 $\boldsymbol{A}$ 和矢量 $\boldsymbol{b}$ 写成分块形式：$\boldsymbol{A}=(\boldsymbol{A}_1,\boldsymbol{A}_2)=\begin{pmatrix}\boldsymbol{O}_{\mu\times n} & \boldsymbol{C}_{\mu\times mn}^{(8,9,10)}\\ \boldsymbol{I}_n & \boldsymbol{C}_{n\times mn}^{(11)}\end{pmatrix}$，$\boldsymbol{b}=\begin{pmatrix}\boldsymbol{p}_{\mu\times mn}^{(8,9,10)}\\ (\boldsymbol{p}-\boldsymbol{d})_{n\times 1}^{(11)}\end{pmatrix}$。其中，$\boldsymbol{I}_n$ 为单位阵，$\boldsymbol{O}_{\mu\times n}$ 为零矩阵，

$\boldsymbol{C}_{\mu\times mn}^{(8,9,10)}$ 和 $\boldsymbol{p}_{\mu\times mn}^{(8,9,10)}$ 为式(8)～(10)对应的系数，$\boldsymbol{I}_n$、$\boldsymbol{C}_{n\times mn}^{(11)}$ 和 $(\boldsymbol{p}-\boldsymbol{d})_{n\times 1}^{(11)}$ 为式(11)对应的系数，$\boldsymbol{p}_{\mu\times mn}^{(8,9,10)}$ 为相关工序加工时间组成的列矢量，$(\boldsymbol{p}-\boldsymbol{d})_{n\times 1}^{(11)}$ 为 $(p_{\mu_1}-d_{\mu_1}),(p_{\mu_2}-d_{\mu_2}),\cdots,(p_{\mu_n}-d_{\mu_n})$ 组成的列矢量。

根据线性规划的对偶原理，式(17)模型的对偶问题模型可以写成如下矩阵标准形式

$$\begin{aligned}\max\quad & \boldsymbol{y}^{\mathrm{T}}\boldsymbol{b}\\ \text{s.t.}\quad & \boldsymbol{y}^{\mathrm{T}}\boldsymbol{A}_1\le\boldsymbol{c}_1^{\mathrm{T}}\\ & \boldsymbol{y}^{\mathrm{T}}\boldsymbol{A}_2\le\boldsymbol{c}_2^{\mathrm{T}}\\ & \boldsymbol{y}^{\mathrm{T}}\ge\boldsymbol{0}^{\mathrm{T}}\end{aligned}\tag{18}$$

式中，对偶变量 $\boldsymbol{y}^{\mathrm{T}}$ 包含 $(\mu+n)$ 个元素，且每一个元素对应式(17)模型或第 1.3 节模型中的约束。同时，使用不同下标的 $F_{*,*}$ 表示 $\boldsymbol{y}^{\mathrm{T}}$ 的元素。譬如，约束 $s_{i_2}-s_{i_1}\ge p_{i_1}$，其对偶变量记为 F_{i_1,i_2} (特殊形式如 $s_{i_1}-p_0\ge s_0$，其对偶变量记为 F_{0,i_1})；约束 $T_{\mu_j}-s_{\mu_j}\ge p_{\mu_j}-d_{\mu_j}$，其对偶变量记为 $F_{\mu_j,0}$。当明确上述关系后，将式(18)模型的标准形式展开为普通形式，即为式(13)～(16)构成的最大成本流问题模型。

证明完毕。

在式(13)～(16)构成的最大成本流问题模型中，当节点1,2,…,*TO* 均满足流量平衡时，虚拟节点0也满足流量平衡。令

$$z=\sum_{\xi:(\xi,0)\in U}F_{\xi,0}-\sum_{\xi:(0,\xi)\in A}F_{0,\xi}\tag{19}$$

遍历所有的弧可得

$$\sum_{i\in N}\sum_{\xi:(\xi,i)\in A\cup\sigma\cup Re\cup U}F_{\xi,i}=\sum_{i\in N}\sum_{\xi:(i,\xi)\in A\cup\sigma\cup Re\cup U}F_{i,\xi}=\sum_{(i_1,i_2)\in A\cup\sigma\cup Re\cup U}F_{i_1,i_2}\tag{20}$$

即有

$$\sum_{i\in N}\sum_{\xi:(\xi,i)\in A\cup\sigma\cup Re\cup U}F_{\xi,i}-\sum_{i\in N}\sum_{\xi:(i,\xi)\in A\cup\sigma\cup Re\cup U}F_{i,\xi}=0\tag{21}$$

因此，如果将式(19)与(15)包含的 *TO* 个约束式相加可得 $z=0$。

2 基于块结构性质的花粉算法

2.1 编码和解码

令种群中第 i 个基于随机数编码的个体为 $\boldsymbol{X}_i=(x_{i,1},x_{i,2},\cdots,x_{i,TO})$，相应的基于工序编码为 $\boldsymbol{\pi}_i=(\pi_{i,1},\pi_{i,2},\cdots,\pi_{i,TO})$，相应的基于析取图工序编码为 $\boldsymbol{\varphi}_i=(\varphi_{i,1},\varphi_{i,2},\cdots,\varphi_{i,TO})$，其中 $\varphi_{i,j}=(\pi_{i,j}-1)\times m\times r+z_{\pi_{i,j}}$，$z_{\pi_{i,j}}$ 为工件 $\pi_{i,j}$ 在 $\boldsymbol{\pi}_i$ 中第 $z_{\pi_{i,j}}$ 次出现。

结合块结构性质和花粉算法的特点，采用基于随机数、工序和析取图工序的混合编码方式，花粉算法求解过程中，采用基于随机数的编码；在块结构上进行邻域工序时采用基于析取图工序的编码方式。不同编码方式之间的相互转化采用扩展 QIAN[17] 提出的 RSOV 规则。如表 2 表示3×2×2 的 RJSSP 编码实例。

表 2 3×2×2 的 **RJSSP** 编码

维数 k	1	2	3	4	5	6	7	8	9	10	11	12
$x_{i,k}$	1.51	2.83	3.25	2.30	1.69	2.57	3.83	0.61	2.40	3.08	0.11	1.03
$y_{i,k}$	4	9	11	6	5	8	12	2	7	10	1	3
$\pi_{i,k}$	1	3	3	2	2	2	3	1	2	3	1	1
$\varphi_{i,k}$	1	9	10	5	6	7	11	2	8	12	3	4

解码方式采用活动化解码的方式，即在不影响已调度的其他工序的前提下尽可能早地调度未调度的工序。由于一个解经活动化解码前后的序列可能不同，当存在工序向前插入时，对应的该个体的序列也应向前调整。为适应块结构的操作，当调度完成之后，将工序按完工时间从小到大排序，得到的新解替换旧解作为当前解输出。

2.2 邻域结构设计

邻域结构的设计和性质分析是组合优化领域的重要研究问题。有效的邻域结构可明显加快算法的收敛速度，并提高整个算法的搜索性能。本节首先给出基于关键路径的块结构定义；其次给出基于块结构的 8 种邻域结构的定义；然后对前 4 种邻域结构的性质进行分析。

为表述方便起见，表 3 给出本节用到的符号表示及其说明。

2.2.1 关键路径与块结构

关键路径定义：基于 MPF(Machine predecessor first, MPF)规则[13]，每个延误的工件从最后一个工序到虚拟节点 0 的最长路径称为该工件的关键路径。

块结构定义：在所有的关键路径中，① 在某一台机器上连续加工的工序的个数大于等于 2，且不全属于一个工件；② 连续加工的工序之间的流量全为正流，则该连续加工的工序称为一个块结构，即在同一台机器上可能有多个块结构。

转折点定义：块结构中的工序 β 满足 $F_{\beta,ISP(\beta)}>0$ 时，即为一个转折点。

基于块结构向前插入操作的定义：工件排列 $\boldsymbol{\varphi}_i=[\varphi_{i,1},\varphi_{i,2},\cdots,\varphi_{i,TO}]$ 的一个块结构中的两个工序 $\varphi_{i,j}=\alpha$，$\varphi_{i,k}=\beta$，且 $j<k$，将 β 插入到 α 之前时，

由于合取弧 A 的约束需要将在当前解中从 $j+1$ 至 k 之间与 β 属于同一工件的所有工序插入到 α 之前，记为 $INS(\alpha,\beta)$。

表 3　本节用到的符号表示及其说明

符号	说明
PJ_i	与工序 i 属于同一个工件的邻接前序工序
SJ_i	与工序 i 属于同一个工件的邻接后序工序
PM_i	与工序 i 在同一个机器上加工的邻接前序工序
SM_i	与工序 i 在同一个机器上加工的邻接后序工序，当不存在定义的工序时，用虚拟工序 0 代替
FJ_i	工序 i 在连接弧 A 上的流量，$FJ_i = F_{i,SJ_i}$
$ISP(\beta)$	在 A 中，与 β 在不同的机器上的工序
$Block_{i,j}$	第 i 个块上的第 j 个工序
$Block\_n_i$	第 i 个块结构所含工序的个数
$Block\_num$	块结构总数
$Block\_Turn_{i,0}$	第 i 个块结构所含转折点的个数
$Block\_Turn_{i,j}$	第 i 个块结构的第 j 个转折点
$\boldsymbol{LBest\_\varphi}$	当前搜索到的最优排列
$Same1(\alpha,\beta)$	判断 α 与 β 之间是否存在与 β 属于同一工件的工序
$\prec$	比较两个序列评价值符号
$\mathrm{Calculate}(\boldsymbol{\varphi})$	计算 $\boldsymbol{\varphi}$ 的评价值
$\mathrm{Satisfy}_i(\alpha,\beta)$	判断 α 与 β 是否满足结构 i 约束
$\mathrm{Satisfy}_i(\alpha,\gamma,\beta)$	判断 α、γ 与 β 是否满足结构 i 约束
$Random[a,b]$	在 a 至 b 随机产生的整数
$insert(\boldsymbol{\varphi},\alpha,\beta)$	基于序列 $\boldsymbol{\varphi}$ 的将 β 插入到 α 之前
$MJ\_Num_{i,1}$	加工工序 i 的机器
$MJ\_Num_{i,2}$	工序 i 在机器 $MJ\_Num_{i,1}$ 加工的位置
$MJ_{i,j}$	机器 i 第 j 个位置加工的工序
$Order\_Num_i$	工序 i 在序列 $\boldsymbol{\varphi}$ 内的位置

定理 2：在当前可行解 $\boldsymbol{\varphi}_i$ 中的两个工序 $\varphi_{i,j}=\alpha$，$\varphi_{i,k}=\beta$，且 $j<k$，采用基于块结构向前插入操作时不会导致不可行解。

证明：如果导致不可行解，即存在有向环，如图 2 所示，表示一个块结构上的 5 个工序，将第 4 个工序插入到第 2 个工序之前。文献[1，13]已证在向前插入操作后只可能在弧 $(4,2)$ 上形成环，但是在按照给出的定义下进行操作时，工序 PJ_4、4 在工序 2、SJ_2 之前，则 SJ_2 和 PJ_4 两工序之间的析取弧方向也会变化，故不会导致有向环。

证明完毕。

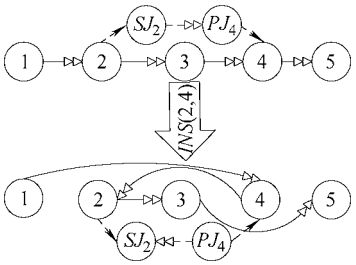

图 2　$INS(2,4)$ 操作

2.2.2　邻域结构的定义

邻域结构是指可直接生成问题解的邻域(即相邻解)的结构。通过在块结构上使用插入操作，可以得到解的相邻解，从而构建邻域结构。本小节设计如下基于块结构的多种邻域结构，用于执行紧凑的邻域搜索。

在连续正流的块结构 $(1,2,3,\cdots,\alpha-1,\alpha,\alpha+1,\cdots,\beta-1,\beta,\beta+1,\cdots,z)$ 内，α 和 β 满足以下几点。

(1) 工序 α 和 β 不属于同一个工件。

(2) β 为转折点。

(3) $\alpha>1$ 时，满足以下约束。

1) 邻域结构 1：$INS^1(\alpha,\beta)$。在当前块结构内，工序 α 和工序 β 间不存在与工序 β 属于相同的工件的工序时，且不满足式(22)，将 β 插入到 α 前。

$$p_\beta\sum_{i=\alpha}^{\beta-1}FJ_i \ge FJ_\beta\sum_{i=\alpha}^{\beta-1}p_i \tag{22}$$

如图 3 所示，表示一个块结构上的 5 个工序，工序 2 和工序 4 符合结构 1 的约束，将工序 4 插入到工序 2 之前，记为 $INS^1(2,4)$。

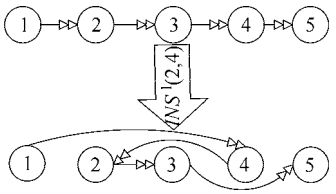

图 3　$INS^1(2,4)$ 操作

下图为 3×2×2 RJSSP 甘特图实例，数据均为随机产生。图 4、5 中工序 5、10、1、11、6、2 是机器 M_1 上的一个块结构，工序 1 和 11 符合结构 1 的约束，图 4 为 $INS^1(1,11)$ 操作之前，图 5 为 $INS^1(1,11)$ 操作之后。

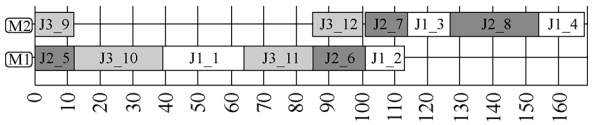

图 4　$INS^1(1,11)$ 操作之前

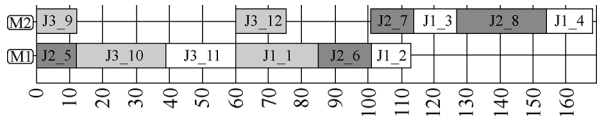

图 5 $INS^{1}(1,11)$ 操作之后

2) 邻域结构 2：$INS^{2}(\alpha,\beta)$。在当前块结构内，工序 α 和工序 β 之间存在与工序 β 属于相同工件的工序 γ，且弧 $(\gamma,\beta)\in(A\cap \mathrm{Re})$，不满足式(23)时，将 γ 和 β 插入到 α 前。

$$p_{\gamma}\sum_{i=\alpha}^{\gamma-1}FJ_i+p_{\beta}\sum_{i=\alpha}^{\beta-1}FJ_i\geq FJ_{\beta}\left(\sum_{i=\alpha}^{\beta-1}p_i-p_{\gamma}\right) \tag{23}$$

如图 6，表示一个块结构上的 5 个工序，工序 PJ_3 和工序 3 符合结构 2 的约束，将工序 PJ_3 和工序 3 插入到工序 2 之前，记为 $INS^{2}(2,3)$。

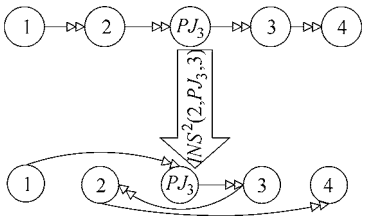

图 6 $INS^{2}(2,PJ_3,3)$ 操作

图 7、8 中工序 1、7、9、2、10、8 是机器 M_1 上的一个块结构，工序 7、9 和 10 符合结构 2 的约束，图 7 为 $INS^{2}(7,9,10)$ 操作之前，图 8 为 $INS^{2}(7,9,10)$ 操作之后。

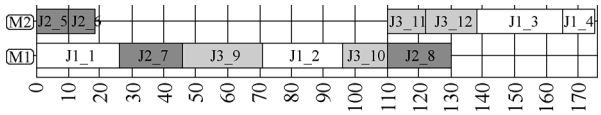

图 7 $INS^{2}(7,9,10)$ 操作之前

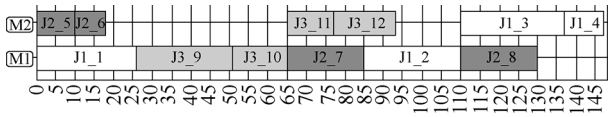

图 8 $INS^{2}(7,9,10)$ 操作之后

3) 在当前块结构内，如果工序 α 和工序 β 之间存在与工序 α 和 β 均不属于同一个工件的工序 γ，且 γ 为转折点，工序 α 和工序 β 之间不存在与工序 β 属于相同的工件的工序且工序 α 和工序 γ 之间不存在与工序 γ 属于相同的工件的工序时：

① 邻域结构 3：$INS^{3}(\alpha,\gamma,\beta)$。不满足式(24)时，将 γ 和 β 插入到 α 前，γ 在 β 之前。

$$p_{\gamma}\sum_{i=\alpha}^{\gamma-1}FJ_i+p_{\beta}\sum_{i=\alpha}^{\beta-1}FJ_i\geq p_{\beta}FJ_{\gamma}+\left(FJ_{\gamma}+FJ_{\beta}\right)\sum_{i=\alpha}^{\gamma-1}p_i+FJ_{\beta}\sum_{i=\gamma+1}^{\beta-1}p_i \tag{24}$$

如图 9 所示，表示一个块结构上的 5 个工序，工序 3 和工序 4 符合结构 3 的约束，将工序 3 和工序 4 插入到工序 2 之前，记为 $INS^{3}(2,3,4)$。

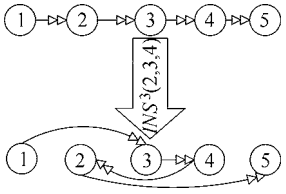

图 9 $INS^{3}(2,3,4)$ 操作

图 10、11 中工序 6、2、10、3、11、8 是机器 M_1 上的一个块结构，工序 3、11 和 8 符合结构 3 的约束，图 10 为 $INS^{3}(3,11,8)$ 操作之前，图 11 为 $INS^{3}(3,11,8)$ 操作之后。

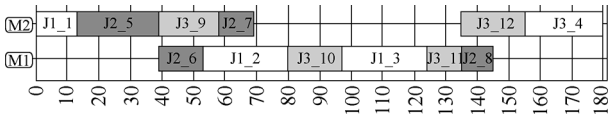

图 10 $INS^{3}(3,11,8)$ 操作之前

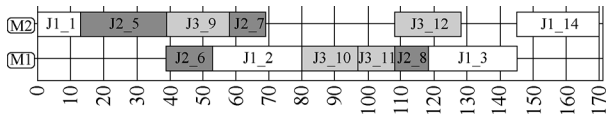

图 11 $INS^{3}(3,11,8)$ 操作之后

② 邻域结构 4：$INS^{4}(\alpha,\gamma,\beta)$。不满足式(25)时，将 γ 和 β 插入到 α 前，β 在 γ 之前。

$$p_{\gamma}\sum_{i=\alpha}^{\gamma-1}FJ_i+p_{\beta}\sum_{i=\alpha}^{\beta-1}FJ_i\geq p_{\gamma}FJ_{\beta}+\left(FJ_{\gamma}+FJ_{\beta}\right)\sum_{i=\alpha}^{\gamma-1}p_i+FJ_{\beta}\sum_{i=\gamma+1}^{\beta-1}p_i \tag{25}$$

如图 12 所示，表示一个块结构上的 6 个工序，工序 2、4、5 符合结构 4 的约束，将工序 4 和工序 5 插入到工序 2 之前，记为 $INS^{4}(2,4,5)$。

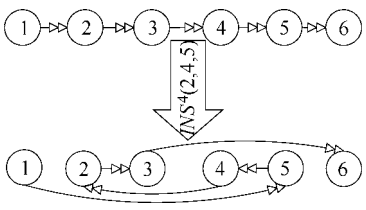

图 12 $INS^{4}(2,4,5)$ 操作

图 13、14 中工序 2、9、6、10、4、7 是机器 M_1 上的一个块结构，工序 6、10 和 4 符合结构 4 的约束，图 13 为 $INS^{4}(6,10,4)$ 操作之前，图 14 为 $INS^{4}(6,10,4)$ 操作之后。

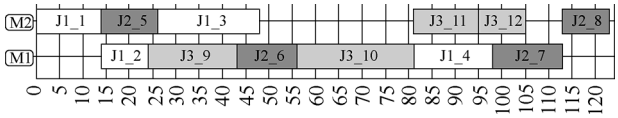

图 13 $INS^{4}(6,10,4)$ 操作之前

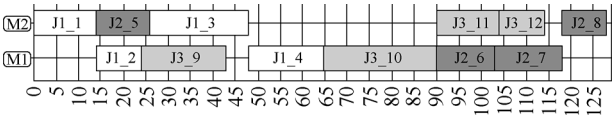

图 14　$INS^4(6,10,4)$ 操作之后

4) 邻域结构 5 至 8：

为在构造局部搜索时适当扩大搜索范围，定义 4 类邻域结构如下：

邻域结构 5：$\alpha=1$ 时，即块首的工序，将块内或块尾的工序向前插入；

邻域结构 6：相邻两个块结构中符合 $F_{i,ISP(i)}>0$ 的工序，分别选取两个工序进行向前插入操作；

邻域结构 7：在同一台机器上，块内工序随机向其所在块结构前插入；

邻域结构 8：在同一台机器上，块结构后的工序随机向块结构内插入；

其中，邻域结构 1～6 在局部搜索中主要使用，邻域结构 7 和 8 只有当邻域结构 1 至 6 不能改进目标值时才使用。其中，邻域结构 2 是 RJSSP 特有的。

2.2.3　邻域结构的性质

定理 3：当采用邻域结构 1 时，RJSSP 的目标函数 *TWT* 不会变小。

定理 3 的证明过程与文献[13]中的类似，这里略过。

定理 4：当采用邻域结构 2 时，RJSSP 的目标函数 *TWT* 不会变小。

证明：令 x_i 和 y_i 为在该块结构中的析取弧 σ 上的流量。假设工序 $[1,2,3,\cdots,z]$ 为一个连续正流的块结构。$INS^2(\alpha,\beta)$ 之前，$x_i=F_{i,SM_i}$，如图 15 所示；$INS^2(\alpha,\beta)$ 之后，$y_i=F_{i,SM_i}$，如图 16 所示。

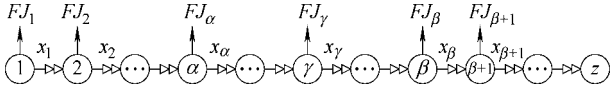

图 15　$INS^2(\alpha,\beta)$ 操作之前

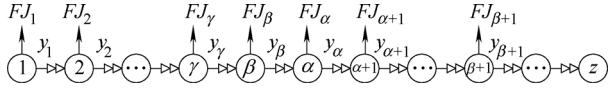

图 16　$INS^2(\alpha,\beta)$ 操作之后

由图 15 可得式(26)

$$\begin{cases} x_{\alpha-1}=FJ_\alpha+x_\alpha \\ x_\alpha=FJ_{\alpha+1}+x_{\alpha+1} \\ x_{\alpha+1}=FJ_{\alpha+2}+x_{\alpha+2} \\ \quad\cdots \\ x_{\beta-1}=FJ_\beta+x_\beta \\ x_\beta=FJ_{\beta+1}+x_{\beta+1} \end{cases} \tag{26}$$

图 16 根据输入输出的流量平衡要求 $y_{\alpha-1}=x_{\alpha-1}$，$y_\alpha=x_\beta$，可得式(27)

$$\begin{cases} y_{\alpha-1}=FJ_\alpha+y_\alpha \\ y_\gamma=FJ_\beta+y_\beta \\ y_\beta=FJ_\alpha+y_\alpha \\ y_\alpha=FJ_{\alpha+1}+y_{\alpha+1} \\ y_{\alpha+1}=FJ_{\alpha+2}+y_{\alpha+2} \\ \quad\cdots \\ y_{\gamma-2}=FJ_{\gamma-1}+y_{\gamma-1} \\ y_{\gamma-1}=FJ_{\gamma+1}+y_{\gamma+1} \\ \quad\cdots \\ y_{\beta-2}=FJ_{\beta-1}+y_{\beta-1} \\ y_{\beta-1}=x_\beta \end{cases} \tag{27}$$

将 $y_{\alpha-1}=x_{\alpha-1}, y_\alpha=x_\beta$ 及交换前的等式代入交换后的等式得

$$\begin{cases} y_{\alpha-1}=x_{\alpha-1} \\ y_\gamma=x_{\alpha-1}-FJ_\gamma \\ y_\beta=x_{\alpha-1}-FJ_\gamma-FJ_\beta \\ y_\alpha=x_\alpha-FJ_\gamma-FJ_\beta \\ y_{\alpha+1}=x_{\alpha+1}-FJ_\gamma-FJ_\beta \\ \quad\cdots \\ y_{\gamma-1}=x_{\gamma-1}-FJ_\gamma-FJ_\beta \\ y_{\gamma+1}=x_{\gamma+1}-FJ_\beta \\ y_{\gamma+2}=x_{\gamma+2}-FJ_\beta \\ \quad\cdots \\ y_{\beta-2}=x_{\beta-2}-FJ_\beta \\ y_{\beta-1}=x_\beta \end{cases} \tag{28}$$

$$\Delta C=C(y)-C(x)=\sum_{i=\alpha}^{\beta}p_iy_i-\sum_{i=\alpha}^{\beta}p_ix_i=$$
$$p_\gamma\sum_{i=\alpha}^{\gamma-1}FJ_i+p_\beta\sum_{i=\alpha}^{\beta-1}FJ_i-$$
$$\left(p_\beta FJ_\gamma+\left(FJ_\gamma+FJ_\beta\right)\sum_{i=\alpha}^{\gamma-1}p_i+FJ_\beta\sum_{i=\gamma+1}^{\beta-1}p_i\right) \tag{29}$$

由弧 $(\gamma,\beta)\in(A\cap\mathrm{Re})$ 知，$\beta=SJ_\gamma$，$F_\gamma=F_{\gamma,SJ_\gamma}=F_{\gamma,\beta}$，由 MPF 规则知 $F_{\gamma,\beta}=0$，故 $F_\gamma=0$，所以

$$\Delta C=p_\gamma\sum_{i=\alpha}^{\gamma-1}FJ_i+p_\beta\sum_{i=\alpha}^{\beta-1}FJ_i-FJ_\beta\left(\sum_{i=\alpha+1}^{\beta-1}p_i-p_\gamma\right) \tag{30}$$

当 $\Delta C\geq 0$ 时，不能改进目标函数 *TWT*。

证明完毕。

定理 5：当采用邻域结构 3 时，RJSSP 的目标函数 *TWT* 不会变小。

定理 5 的证明和定理 4 类似，这里略过。

定理 6：当采用邻域结构 4 时，RJSSP 的目标函数 *TWT* 不会变小。

定理 6 的证明和定理 4 类似，这里略过。

依据定理 3 至定理 6，在构造基于邻域结构的局部搜索时，可避免对无效区域的搜索，从而提高算法效率。

2.3 基于邻域结构的局部搜索

基于块结构的邻域结构在使用之前，要首先计算出对偶问题模型中的解，即在析取图模型中每一条弧上的流量 F，然后在根据流量 F 确定块结构。

获得流量 F 的算法与文献[13]相同，这里略过。

确定具有正流特性块结构的算法在文献[13]的基础上需再计算以下内容：***Block_num***、***Block_n***、***Block***、***Block_Turn***。

2.4 基于邻域结构的花粉算法

花粉算法是由 YANG[19]在 2012 年提出的一种新型的元启发式群智能算法，本文首次将其应用到求解车间调度问题。第 2.1 节的 RSOV 规则和本节的 FPA 结合，即可用于求解 RJSSP。在本文提出 FPA_BSP 中，采用 FPA 对问题解空间进行全局搜索，同时采用如图 17 所示的局部搜索，局部搜索的详细步骤在表 4 中给出。

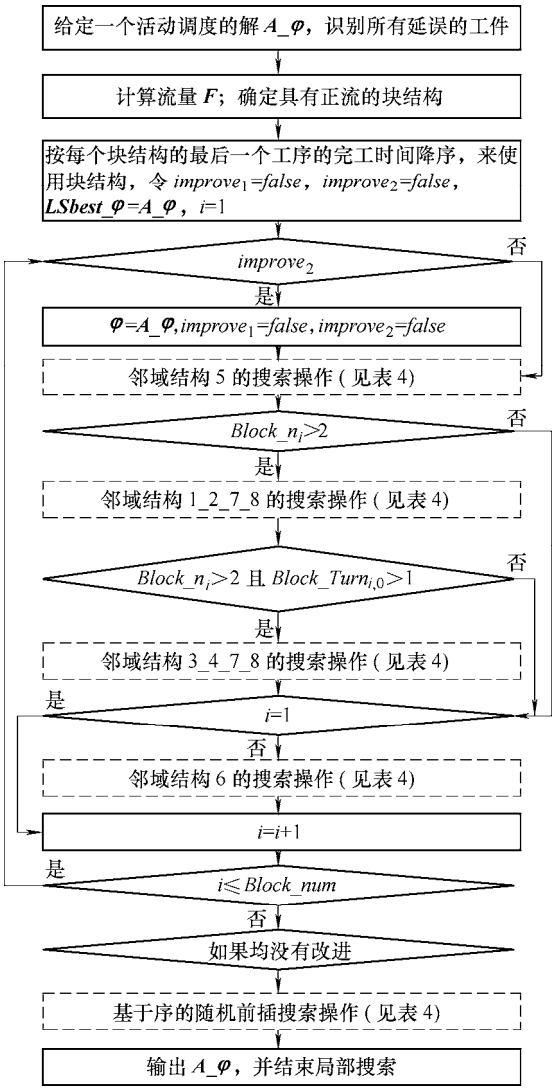

图 17 局部搜索流程图

令 $\boldsymbol{X}_i^t$ 和 $\boldsymbol{X}_i^{t+1}$ 分别为第 t 代和第 $t+1$ 代的第 i 个个体；***G_Best*** 为全局最优个体；L 为步长，$\Gamma(\lambda)$ 为标准的伽马函数；$\boldsymbol{X}_j^t$、$\boldsymbol{X}_k^t$ 为第 t 代中不同的两个个体；P、ε 为 $[0,1]$ 按均匀分布产生的随机数；U 为按均值为 0 方差为 σ^2 的正态分布产生的随机数；V 为按标准正态分布产生的随机数。

基于邻域结构的花粉算法基本步骤如下。

步骤 1：种群规模 *popsize*；机器数 m、工件数 n、重入次数 r；每道工序的加工时间；工件在机器上的加工次序约束；每个工件的交货期及延误权重。对种群中 *popsize* 个花粉粒子初始化，确定此时种群最优个体 ***G_Best***，对 ***G_Best*** 进行局部搜索，定义转换概率 P，当前迭代次数 $t=1$，及最大迭代次数 Max*Iters*。

步骤 2：从种群中随机选择一个花粉粒子进行授粉：随机产生一个 *Rand*，若 $Rand<P$，则按照式(31)进行全局授粉；否则，按照式(33)进行局部授粉。评价新解，若新解更好，则替换旧解。找到当前的最优解 ***G_Best***。

$$\boldsymbol{X}_i^{t+1}=\boldsymbol{X}_i^t+\gamma\boldsymbol{L}(\lambda)\left(\boldsymbol{G\_Best}-\boldsymbol{X}_i^t\right) \tag{31}$$

$$L\sim\frac{\lambda\Gamma(\lambda)\sin(\pi\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}}\qquad s\gg s_0>0 \tag{32}$$

$$\boldsymbol{X}_i^{t+1}=\boldsymbol{X}_i^t+\varepsilon\left(\boldsymbol{X}_j^t-\boldsymbol{X}_k^t\right) \tag{33}$$

$$s=\frac{U}{|V|^{1/\lambda}}\qquad U\sim N\left(0,\sigma^2\right)\qquad V\sim N(0,1) \tag{34}$$

$$\sigma^2=\left[\frac{\Gamma(1+\lambda)}{\lambda\Gamma\left((1+\lambda)/2\right)}\cdot\frac{\sin(\pi\lambda/2)}{2^{(\lambda-1)/2}}\right]^{1/\lambda} \tag{35}$$

步骤 3：探索阶段，令 $es=0$；

步骤 3.1：令 $\boldsymbol{\varphi}_0$ 为 ***G_Best*** 基于扩展的 RSOV 规则的基于析取图操作的排序。

步骤 3.2：扰动阶段，令 $kk=0$，$\boldsymbol{\varphi}_1=\boldsymbol{\varphi}_0$，扰动次数 $ks=5$。

步骤 3.2.1：随机选择两个不同位置 $p1$ 和 $p2$，$p1<p2$，采用基于序的随机前插操作，$\boldsymbol{\varphi}_1=insert(\boldsymbol{\varphi}_1,p1,p2)$。

步骤 3.2.2：令 $kk=kk+1$，如果 $kk\le ks$，转步骤 3.2.1；

步骤 3.3：令 $kt=0$。

步骤 3.4：令 $\boldsymbol{\varphi}_2=\boldsymbol{\varphi}_1$，评价 $\boldsymbol{\varphi}_2$ 并对其进行局部搜索，如果 $\boldsymbol{\varphi}_2$ 优于 $\boldsymbol{\varphi}_1$，则 $\boldsymbol{\varphi}_1=\boldsymbol{\varphi}_2$ 否则 $kt=kt+1$。

步骤 3.5：如果 $kt<1$，返回步骤 3.4。

步骤 3.6：如果 $\boldsymbol{\varphi}_1$ 优于 $\boldsymbol{\varphi}_0$，则 $\boldsymbol{\varphi}_0=\boldsymbol{\varphi}_1$ 并退出探索阶段，否则令 $es=es+1$。

表 4　基于邻域结构的搜索操作详细步骤

基于邻域结构的搜索操作	步骤
“邻域结构 1_2_7_8 的搜索操作”	步骤 1：令 $j=1$。 步骤 2：令 $\beta=Block_{i,Block\_Turn_{i,j}}$，$k=2$。 步骤 3：令 $\alpha=Block_{i,k}$，如果 $\text{Satisfy}_1(\alpha,\beta)=false$ 则转步骤 4，否则 $\boldsymbol{\varphi}=\boldsymbol{A\_\varphi}$，$INS^1(\alpha,\beta)$，$\text{Calculate}(\boldsymbol{\varphi})$；如果 $\boldsymbol{\varphi}\prec\boldsymbol{LSbest\_\varphi}$，则 $\boldsymbol{LSbest\_\varphi}=\boldsymbol{\varphi}$，$improve_1=true$，转步骤 5 否则执行下面“邻域结构 7_8 的搜索操作”；如果 $\text{Satisfy}_2(\alpha,\beta)=false$ 则转步骤 4 否则 $\boldsymbol{\varphi}=\boldsymbol{A\_\varphi}$，$INS^2(\alpha,\beta)$，$\text{Calculate}(\boldsymbol{\varphi})$；如果 $\boldsymbol{\varphi}\prec\boldsymbol{LSbest\_\varphi}$，则 $\boldsymbol{LSbest\_\varphi}=\boldsymbol{\varphi}$，$improve_1=true$，转步骤 5 否则执行下面“邻域结构 7_8 的搜索操作”。 步骤 4：令 $k=k+1$，如果 $k\le Block\_Turn_{i,0}-1$，则转步骤 3。 步骤 5：令 $j=j+1$，如果 $j\le Block\_Turn_{i,0}$，则转步骤 2。
“邻域结构 3_4_7_8 的搜索操作”	步骤 1：令 $x=1$。 步骤 2：令 $y=2$，$\gamma=Block_{i,Block\_Turn_{i,x}}$，$\beta=Block_{i,Block\_Turn_{i,x+1}}$。 步骤 3：令 $\alpha=Block_{i,y}$，如果 $\text{Satisfy}_3(\alpha,\gamma,\beta)=false$ 则转步骤 4 否则 $\boldsymbol{\varphi}=\boldsymbol{A\_\varphi}$，$INS^3(\alpha,\gamma,\beta)$，$\text{Calculate}(\boldsymbol{\varphi})$；如果 $\boldsymbol{\varphi}\prec\boldsymbol{LSbest\_\varphi}$，则 $\boldsymbol{LSbest\_\varphi}=\boldsymbol{\varphi}$，$improve_1=true$，转步骤 5 否则执行下面“邻域结构 7_8 的搜索操作”；如果 $\text{Satisfy}_4(\alpha,\gamma,\beta)=false$ 则转步骤 4 否则 $\boldsymbol{\varphi}=\boldsymbol{A\_\varphi}$，$INS^4(\alpha,\gamma,\beta)$，$\text{Calculate}(\boldsymbol{\varphi})$；如果 $\boldsymbol{\varphi}\prec\boldsymbol{LSbest\_\varphi}$，则 $\boldsymbol{LSbest\_\varphi}=\boldsymbol{\varphi}$，$improve_1=true$，转步骤 5 否则执行下面“邻域结构 7_8 的搜索操作”。 步骤 4：令 $y=y+1$，如果 $y\le Block\_Turn_{i,x}-1$，则转步骤 3。 步骤 5：令 $x=x+1$，如果 $x\le Block\_Turn_{i,0}-1$，则转步骤 2。
“邻域结构 5 的搜索操作”	步骤 1：令 $j=1$。 步骤 2：令 $\beta=Block_{i,j}$，如果 $Same1(\alpha,\beta)=true$，则 $\boldsymbol{\varphi}=\boldsymbol{A\_\varphi}$，$INS(\alpha,\beta)$，$\text{Calculate}(\boldsymbol{\varphi})$；如果 $\boldsymbol{\varphi}\prec\boldsymbol{LSbest\_\varphi}$，则 $\boldsymbol{LSbest\_\varphi}=\boldsymbol{\varphi}$，$improve_1=true$。 步骤 3：令 $j=j+1$，如果 $j\le Block\_n_i$ 则转步骤 2。
“邻域结构 6 的搜索操作”	步骤 1：令 $x=1$。 步骤 2：令 $\alpha=Block_{i,Block\_Turn_{i,x}}$，$y=1$。 步骤 3：令 $\beta=Block_{i-1,Block\_Turn_{i-1,y}}$，$\boldsymbol{\varphi}=\boldsymbol{A\_\varphi}$，在 $\boldsymbol{\varphi}$ 内如果 α 在 β 之后，则互换；$INS(\alpha,\beta)$，$\text{Calculate}(\boldsymbol{\varphi})$；如果 $\boldsymbol{\varphi}\prec\boldsymbol{LSbest\_\varphi}$，则 $\boldsymbol{LSbest\_\varphi}=\boldsymbol{\varphi}$，$improve_2=true$。 步骤 4：令 $y=y+1$，如果 $y\le Block\_Turn_{i-1,0}$，则转步骤 3。 步骤 5：令 $x=x+1$，如果 $x\le Block\_Turn_{i,0}$，则转步骤 2。
“邻域结构 7_8 的搜索操作”	步骤 1：如果 $MJ\_Num_{\alpha,2}<m\times r$ 或 $Order\_Num_{\alpha}<TO$ 时，转步骤 2，否则转步骤 3。 步骤 2：邻域结构 7：令 $\boldsymbol{\varphi}=\boldsymbol{A\_\varphi}$，$x=Random\left[MJ\_Num_{\alpha,2},m\times r\right]$，$insert\left(\boldsymbol{\varphi},MJ_{MJ\_Num_{\alpha,1},x},\alpha\right)$，$\text{Calculate}(\boldsymbol{\varphi})$，如果 $\boldsymbol{\varphi}\succ\boldsymbol{LSbest\_\varphi}$ 则转步骤 3，否则 $\boldsymbol{LSbest\_\varphi}=\boldsymbol{\varphi}$，$improve_1=true$。 步骤 3：邻域结构 8：令 $x=Random\left[1,MJ\_Num_{\alpha,2}\right]$，$\boldsymbol{\varphi}=\boldsymbol{A\_\varphi}$，$insert\left(\boldsymbol{\varphi},MJ_{MJ\_Num_{\alpha,1},x},\alpha\right)$，$\text{Calculate}(\boldsymbol{\varphi})$，如果 $\boldsymbol{\varphi}\prec\boldsymbol{LSbest\_\varphi}$，则 $\boldsymbol{LSbest\_\varphi}=\boldsymbol{\varphi}$，$improve_1=true$。
“基于序的随机前插搜索操作”	步骤 1：令 $x=1$。 步骤 2：扰动阶段：令 $y=5$。 步骤 2.1：$z=Random[1,TO-y]$，$\boldsymbol{\varphi}=insert(\boldsymbol{\varphi},z,z+y)$。 步骤 2.2：令 $y=y+1$，如果 $y<11$，返回步骤 2.1。 步骤 3：$\text{Calculate}(\boldsymbol{\varphi})$，如果 $\boldsymbol{\varphi}\prec\boldsymbol{A\_\varphi}$，则 $\boldsymbol{A\_\varphi}=\boldsymbol{\varphi}$，并结束。 步骤 4：令 $x=x+1$，如果 $x<6$，令 $\boldsymbol{\varphi}=\boldsymbol{A\_\varphi}$，转步骤 2。

步骤 3.7：如果 $es=1$，令 $\boldsymbol{\varphi}_1=\boldsymbol{\varphi}_0$，评价 $\boldsymbol{\varphi}_1$ 并对其进行局部搜索(见图 17 和表 4)，如果 $\boldsymbol{\varphi}_1$ 优于 $\boldsymbol{\varphi}_0$，则 $\boldsymbol{\varphi}_0=\boldsymbol{\varphi}_1$ 并退出探索阶段。

步骤 3.8：如果 $es\le 5$ 则转步骤 3.1。

步骤 4：通过扩展的 RSOV 规则将 $\boldsymbol{\varphi}_0$ 转换为 ***G_Best***。

步骤 5：令 $t=t+1$，如果 $t\le \mathrm{Max}Iters$，转步骤 2，否则输出全局最优个体 ***G_Best***，算法结束。

由上述算法步骤可知，花粉算法用于对问题解空间进行全局搜索并发现存在优质解的区域，同时融入不同邻域结构的局部搜索用于对优质解区域执行进一步细致搜索，使得 FPA_BSP 在全局和局部搜索间达到合理平衡，有望成为求解本文问题的有效算法。第 3 节将通过仿真试验和算法比较验证所提算法的有效性。

3 试验结果与分析

测试问题采用随机方式生成。对于不同的问题，n 个工件随机在 m 台机器上加工，每个工件可重复加工 r 次，每道工序的加工时间为 $[1,99]$ 按均匀分布随机产生的整数，每个工件的交货期和工件的总的加工时间相关，$d_j=\left\lfloor \mu\times f\times\sum_{i\in O_j}p_j \right\rfloor$，$\mu\sim U\left[1,1.1\times\max\left\{1,n/(m\times r)\right\}\right]$，$O_j$ 为工件 j 所有工序的集合，$f\in\{1.1,1.3,1.5\}$ 为生成交货期时的松紧程度，每个工件的延误权重为 $[1,10]$ 按均匀分布随机产生的整数。用于测试的 $n\times m\times r$ 组合包含：$10\times10\times3$，$20\times5\times3$，$10\times20\times3$，$20\times10\times3$，$20\times15\times3$，$50\times6\times3$，$20\times20\times3$，$40\times10\times3$，$50\times10\times3$，$100\times5\times3$。由于本文算法针对不同的问题每一代运行时间相差较大，故给定相应问题的运行代数：20，20，15，20，15，10，15，15，10，5。

为了测试所提出的 FPA_BSP 的性能，将 FPA_BSP 与 NIMGA[21]和 B-SA[13]这两个文献中的有效算法进行对比。NIMGA 在与多种有效算法比较时性能占优。B-SA 与 FPA_BSP 类似，同样利用了基于块结构的邻域性质构造搜索邻域，并取得良好性能。但是，B-SA 设计和采用了一种随机交换和三种基于性质的邻域结构。FPA_BSP 的参数设置为：种群规模 $popsize=5$；$\lambda=1.5$；转换概率 $P=0.8$。需指出的是，经测试，FPA_BSP 采用花粉算法执行适度的全局搜索，确保算法整体搜索具有一定的分散性，然后把更多时间用于执行所提出的局部搜索，可使算法的性能更好。这也侧面验证了所构造邻域结构的有效性。通过对种群规模进行的测试，发现 5 个体已能确保算法具有良好性能。因此，FPA_BSP 的种群规模设置为 5。

所有算法的测试程序均由 Delphi10.2 编程实现，操作系统为 Win10，处理器为 Intel Core I7-7700k 4.20 GHz，内存为 8 GB。FPA_BSP 运行相应的代数，对比算法对每个测试问题均在与 FPA_BSP 相同的运行时间下独立运行 20 次，其中 *min*、*max*、*avg* 和 *std* 分别为最小值、最大值、平均值和标准差。表 5 是不同问题的块结构个数统计表，随机生成一个个体，重复 20 次后块结构个数平均值，平均值越大，邻域结构越多。每个问题对应的最优结果用粗体表示，测试结果如表 6～8 所示。

表 5 块结构个数统计表

$n\times m\times r$	$f=1.1$	$f=1.3$	$f=1.5$
$10\times10\times3$	17	17	10
$20\times5\times3$	12	13	15
$10\times20\times3$	20	19	18
$20\times10\times3$	25	25	24
$20\times15\times3$	31	29	30
$50\times6\times3$	28	24	22
$20\times20\times3$	36	36	33
$40\times10\times3$	33	30	34
$50\times10\times3$	37	37	36
$100\times5\times3$	34	30	33

由表 6～8 可知，对于本文采用的 *min*、*max* 和 *avg* 指标，除 $f=1.1$ 时的 $10\times10\times3$ 问题、$f=1.3$ 时

表 6 $f=1.1$ 时的比较结果

问题规模	FPA_BSP				NIMGA[21]				B-SA[13]			
$n\times m\times r$	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*
$10\times10\times3$	**25 138**	34 508	30 771.70	2 423.02	27 929	**33 940**	**30 468.65**	**1 506.62**	35 370	44 779	42 468.80	2 226.38
$20\times5\times3$	**142 222**	164 019	**150 040.90**	5 368.40	152 049	**163 522**	157 332.35	3 432.56	185 937	198 242	193 925.65	**2 680.14**
$10\times20\times3$	**17 528**	**27 395**	**21 465.80**	2 304.49	20 901	28 645	25 878.45	1 956.60	32 062	36 018	34 060.50	**953.40**
$20\times10\times3$	**157 350**	**176 907**	**169 384.80**	5 255.19	180 108	196 874	188 907.00	3 821.75	205 669	214 035	210 598.15	**2 480.22**
$20\times15\times3$	**155 740**	**180 419**	**167 721.20**	6 976.56	167 975	198 200	183 859.30	7 127.06	212 084	220 623	217 009.85	**2 722.64**
$50\times6\times3$	**716 609**	**833 779**	**768 747.15**	32 629.98	822 315	899 402	854 320.90	19 906.94	1 122 045	1 193 950	1 161 696.20	**19 329.35**
$20\times20\times3$	**144 646**	**162 640**	**152 249.65**	5 378.45	170 787	190 376	178 362.30	5 239.37	194 171	203 247	198 294.35	**2 584.29**
$40\times10\times3$	**670 987**	**729 863**	**706 262.25**	16 798.61	795 859	860 791	838 801.00	15 850.33	933 843	954 258	942 598.40	**5 255.11**
$50\times10\times3$	**838 895**	**956 943**	**900 552.65**	31 132.62	992 583	1 074 426	1 033 015.05	21 355.29	1 211 533	1 275 970	1 245 943.20	**15 109.97**
$100\times5\times3$	**2 243 404**	**2 890 404**	**2 588 892.30**	161 205.19	2 719 616	3 051 141	2 864 783.70	**100 638.10**	4 649 756	5 064 101	4 820 790.40	110 108.90

表 7　$f=1.3$ 时的比较结果

问题规模	FPA_BSP				NIMGA[21]				B-SA[13]			
$n\times m\times r$	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*
10×10×3	**21 717**	26 858	251 14.15	1 245.61	22 201	**26 276**	**24 140.55**	**1 115.34**	33 243	40 971	38 108.15	1 778.18
20×5×3	82 278	111 250	98 680.90	7 269.16	**71 859**	**85 342**	**78 976.70**	**3 511.35**	125 606	147 548	134 116.30	5 151.69
10×20×3	**16 641**	**24 735**	**20 668.85**	1 846.85	19 070	26 498	23 728.50	1 634.96	30 032	36 924	34 008.90	**1 505.42**
20×10×3	**112 057**	**133 161**	**122 994.30**	4 563.30	123 466	138 505	131 437.90	3 589.51	163 096	172 663	168 511.75	**2 761.79**
20×15×3	**150 950**	**173 779**	**164 452.05**	6 457.45	180 136	195 067	188 699.70	3 616.60	205 324	219 830	214 498.85	**3 509.77**
50×6×3	**669 879**	**840 324**	**748 248.55**	42 095.31	773 577	854 912	812 217.55	23 156.44	1 102 346	1 145 783	1 120 503.65	**11 477.95**
20×20×3	**153 574**	**180 570**	**168 638.10**	7 925.56	190 313	206 378	198 922.25	**3 805.49**	215 892	231 870	225 065.25	4 051.17
40×10×3	**621 542**	**682 641**	**657 698.25**	12 663.40	733 678	797 455	771 397.95	17 170.97	861 339	905 635	881 351.60	**11 596.55**
50×10×3	**693 524**	**789 653**	**741 979.15**	22 833.46	858 470	967 982	916 583.75	25 449.57	1 089 126	1 117 706	1 106 384.25	**7 148.55**
100×5×3	**2 745 712**	**3 272 574**	**3 001 392.25**	137 103.61	3 072 452	3 346 782	3 251 248.05	**68 784.90**	5 214 664	5 586 304	5 437 483.80	109 239.26

表 8　$f=1.5$ 时的比较结果

问题规模	FPA_BSP				NIMGA[21]				B-SA[13]			
$n\times m\times r$	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*	*min*	*max*	*avg*	*std*
10×10×3	**0**	**0**	**0.00**	**0.00**	**0**	1490	429.25	479.97	1 780	5 150	3 790.75	909.81
20×5×3	88 570	111 559	100 613.45	5 559.98	**86 177**	**98 232**	**91 902.35**	2 974.13	143 317	149 044	145 913.60	**1 740.47**
10×20×3	**1 915**	**3 915**	**2 823.45**	**584.82**	3 461	5 740	4 556.75	623.30	12 273	16 097	13 917.30	1 017.58
20×10×3	**68 251**	**90 355**	**76 612.40**	5 696.19	75 876	90 598	84 466.20	4 101.77	107 566	119 506	112 814.45	**2 431.14**
20×15×3	**80 302**	**114 593**	**96 969.30**	9 322.87	107 184	122 091	114 942.85	4 813.10	140 622	154 199	147 359.05	**3 806.09**
50×6×3	**546 002**	**663 430**	**604 393.35**	27 291.17	627 668	695 053	658 547.65	20 281.87	931 296	1 016 077	959 337.60	**19 824.28**
20×20×3	**76 914**	**92 642**	**84 598.05**	4 760.51	100 370	122 034	109 921.35	5 683.08	138 079	148 252	143 957.85	**2 759.42**
40×10×3	**502 054**	**590 335**	**545 054.40**	21 223.76	623 874	658 711	641 052.00	7 951.84	696 120	729 754	717 011.15	**7 942.65**
50×10×3	**997 898**	**1 154 605**	**1 084 731.85**	48 203.55	1 238 120	1 321 218	1 283 937.15	21 788.05	1 414 198	1 465 962	1 437 017.15	**14 208.61**
100×5×3	**2 325 190**	**2 729 482**	**2 536 590.50**	105 535.55	2 665 553	2 992 527	2 853 333.80	94 767.62	4 427 832	4 770 639	4 631 929.65	**92 225.93**

的 10×10×3 问题和 20×5×3 问题及 $f=1.5$ 时的 20×5×3 问题外，FPA_BSP 在其他所有问题上均为最优。上述的 10×10×3 和 20×5×3 均属于小规模问题。同时，$f=1.5$ 时 10×10×3 问题的 4 个指标均为 0，可知 FPA_BSP 算法每次运行均找到该问题的最优解。由于计算流量和统计块结构需花费一定时间，且从表 5 可知小规模问题块结构数量也最少，故在部分小规模问题上本文算法不如 NIMGA。当问题规模增大，当块结构数量随之变多时，FPA_BSP 性能明显优于 NIMGA。此外，FPA_BSP 在所有问题上的性能均明显优于同样采用块结构邻域的 B-SA，这表明更多地挖掘和利用邻域结构性质，可在相同时间内搜索更多有价值的解空间子区域，从而能有效提高算法性能。

4　结论

针对以最小化总加权延误时间(Total weighted tardiness, TWT)为目标的可重入作业车间调度问题(Reentrant job shop scheduling problem, RJSSP)，提出一种基于块结构性质的花粉算法(Flower pollination algorithm based on block structure properties, FPA_BSP)进行有效求解。这是首次将基于 FPA 的算法用于求解此类问题，也是首次使用 FPA 解决车间调度问题。具体结论如下。

(1) 建立 RJSSP 的对偶模型，并利用该模型问题特性得到关键路径上的块结构内部性质。

(2) 利用块结构性质，提出 4 种可排除块结构内部无效插入操作的新型邻域结构，从而构造出紧凑、高效的搜索邻域。

(3) 提出融合 FPA 和多邻域局部搜索的混合 FPA_BSP，使得算法的全局搜索和局部搜索均具有较好的指导性和方向性。未来的研究工作是针对不确定多目标的 RJSSP，挖掘有效邻域结构，并设计有效的多目标 FPA_BSP 进行求解。

参 考 文 献

[1] 潘全科，王凌，高亮，等. 基于差分进化与块结构邻域的作业车间调度优化[J]. 机械工程学报，2010，46(22)：182-188.
PAN Quanke，WANG Ling，GAO Liang，et al. Differential evolution algorithm based on blocks on critical path for job shop scheduling problems[J]. Journal of Mechanical Engineering，2010，46(22)：182-188.

[2] GAREY M R, JOHNSON D S, SETHI R. The complexity of flowshop and jobshop scheduling[J]. Mathematics of Operations Research, 1976, 1(2)：117-129.

[3] 唐国春，张峰，罗守成，等. 现代排序论[M]. 上海：上海科学普及出版社，2003.
TANG Guochun, ZHANG Feng, LUO Shoucheng, et al. Modern scheduling theory[M]. Shanghai ： Shanghai Popular Science Press, 2003.

[4] VAN LAARHOVEN P J M, AARTS E H L, LENSTRA J K. Job shop scheduling by simulated annealing[J]. Operations Research, 1988, 40(1)：113-125.

[5] BALAS E. Machine sequencing via disjunctive graphs: an implicit enumeration algorithm[J]. Operations Research, 1969, 17(17)：941-957.

[6] MATSUO H, SUH C J, SULLIVAN R S. A controlled search simulated annealing method for the general job-shop scheduling problem[J]. Annals of Operations Research, 1988, 21(1)：85-108.

[7] GRABOWSKI J, NOWICKI E, SMUTNICKI C. Block algorithm for scheduling operations in a job-shop system[J]. Przeglad Statystyczny, 1988, 35(1)：67-80.

[8] GRABOWSKI J, NOWICKI E, ZDRZALKA S. A block approach for single-machine scheduling with release dates and due dates[J]. European Journal of Operational Research, 1986, 26(2)：278-285.

[9] DELL'AMICO M, TRUBIAN M. Applying tabu search to the job-shop scheduling problem[J]. Annals of Operations Research, 1993, 41(3)：231-252.

[10] NOWICKI E, SMUTNICKI C. A fast taboo search algorithm for the job shop problem[J]. Management Science, 1996, 42(6)：797-813.

[11] BALAS E, VAZACOPOULOS A. Guided local search with shifting bottleneck for job shop scheduling[J]. Management Science, 1998, 44(2)：262-275.

[12] ZHANG C Y, LI P G, GUAN Z L, et al. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem[J]. Computers & Operations Research, 2007, 34(11)：3229-3242.

[13] ZHANG R, WU C. A simulated annealing algorithm based on block properties for the job shop scheduling problem with total weighted tardiness objective[J]. Computers & Operations Research, 2011, 38(5)：854-867.

[14] KUHPFAHL J, BIERWIRTH C. A study on local search neighborhoods for the job shop scheduling problem with total weighted tardiness objective[J]. Computers & Operations Research, 2016, 66：44-57.

[15] 赵诗奎. 基于新型邻域结构的混合算法求解作业车间调度[J]. 机械工程学报，2016，52(9)：141-151.
ZHAO Shikui. A hybrid algorithm with a new neighborhood structure for the job shop scheduling problem[J]. Journal of Mechanical Engineering, 2016, 52(9)：141-151.

[16] TOPALOGLU S, KILINCLI G. A modified shifting bottleneck heuristic for the reentrant job shop scheduling problem with makespan minimization[J]. International Journal of Advanced Manufacturing Technology, 2009, 44(7-8)：781-794.

[17] QIAN B, LI Z H, HU R, et al. A hybrid differential evolution algorithm for the multi-objective reentrant job-shop scheduling problem[C]//10th IEEE International Conference on Control and Automation (ICCA), Hangzhou, 2013: 485-489.

[18] CHEN S F, QIAN B, LIU B, et al. Bayesian statistical inference-based estimation of distribution algorithm for the re-entrant job-shop scheduling problem with sequence-dependent setup times[C]// 10th International Conference on Intelligent Computing (ICIC), Taiyuan, 2014. Lecture Notes in Computer Science, Springer, Cham, 2014, (8589)：686-696.

[19] YANG X S. Flower pollination algorithm for global optimization[C]// International Conference on Unconventional Computation and Natural Computation. Springer-Verlag, 2012：240-249.

[20] WENNINK M. Algorithmic support for automated planning boards[D]. Eindhoven, Netherlands：Technische Universiteit Eindhoven, 1995.

[21] KURDI M. An effective new island model genetic algorithm for job shop scheduling problem[J]. Computers & Operations Research, 2016, 67：132-142.

作者简介：孙在省，男，1993 年出生。主要研究方向为优化调度理论与方法。
E-mail：szx_1010@163.com
钱斌(通信作者)，男，1976 年出生，教授，博士。主要研究方向为优化调度理论与方法、智能优化方法。
E-mail：bin.qian@vip.163.com